MyTone: compositor musical con algoritmos genéticos

Claudia Lertora Ginés
Paloma López de Arenosa Barbeito
Joanna Zevallos Rodríguez

PROYECTO DE FACULTAD DE INFORMÁTICA
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

UNIVERSIDAD COMPLUTENSE DE MADRID

TRABAJO DE FIN DE CARRERA EN INGENIERÍA INFORMÁTICA

20/06/2013

Director:

Jaime Sánchez Hernández

Autorización

Claudia Lertora, Paloma López de Arenosa y Joanna Zevallos autorizan a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, los contenidos audiovisuales incluso si incluyen imágenes de los autores, la documentación y/o el prototipo desarrollado.

Claudia Lertora Paloma López de Arenosa Joanna Zevallos

a 20 de Junio de 2013

Índice general

Resumen

Desde hace tiempo se vienen utilizando los algoritmos genéticos como herramienta para la creación de composiciones musicales. A continuación se presenta un sistema capaz de asistir al usuario durante el proceso de composición de música. En este caso, el usuario juega un papel clave sobre los elementos generados puesto que es el encargado de evaluar la aptitud de los motivos musicales derivados. Componer música con MyTone implica la edición de fragmentos musicales, la generación de nuevas piezas por medio de un algoritmo inspirado en los algoritmos genéticos y la creación de una pieza musical a partir de muestras musicales más cortas. El proyecto proporciona una interfaz gráfica sencilla e intuitiva que concede al usuario un control total sobre las fases de edición y composición, así como la posibilidad de manipular los parámetros internos del algoritmo. El objetivo de este software es proporcionar un asistente de composición apto para cualquier usuario, de forma que independientemente de sus conocimientos musicales pueda obtener melodías relativamente agradables y complejas.

Palabras Clave: algoritmos genéticos, asistente de composición musical, notación ABC, MIDI.

Genetic algorithms have been used for some time as a tool to create musical compositions. Detailed below is a system capable of assisting a user throughout the process of composing music. In this case, the user plays a key role in the generated results as he or she is the one in charge of evaluating the adequacy of the generated musical motives. Composing music with MyTone involves editing musical pieces, generating musical fragments through the means of an algorithm inspired by genetic algorithms, and creating a tune from smaller sample tunes. The project provides a simple and intuitive graphic interface for the user, granting him or her full control over the editing and composing phases, and granting the possibility of manipulating the parameters within the algorithm. The goal of this software is to provide an aid for composing music, fit for any type of user regardless of his or her musical knowledge so they may be able to obtain relatively pleasant and complex melodies.

Keywords: genetic algorithms, music composer assistant, ABC music notation, MIDI.

Capítulo 1

Introducción

1.1. Un poco de MyTone

MyTone es un asistente de composición musical inspirado en el concepto de los algoritmos genéticos. Se trata de un programa capaz de generar fragmentos musicales o "piezas" a partir de otras basándose en dicho concepto.

Está pensado para que un público no necesariamente experto en música pueda aprender y divertirse usando la composición y experimentando con ella.

A continuación se describirán las nociones básicas de la notación ABC, se introducirá el concepto de los algoritmos genéticos y en qué consisten, debido a la importancia que han tenido a la hora de desarrollar el algoritmo generador de música. Además se mencionarán las aplicaciones informáticas musicales implementadas con algoritmos genéticos y sus logros junto con las motivaciones y los objetivos propuestos para este proyecto.

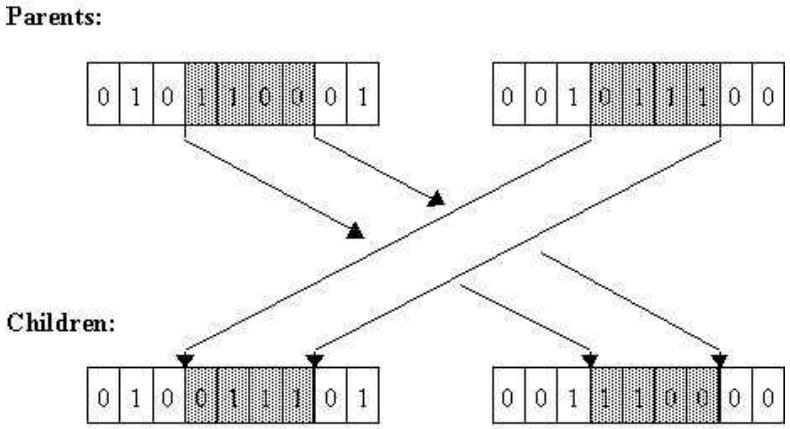
1.2. Introducción a los Algoritmos Genéticos

Los algoritmos genéticos son algoritmos de búsqueda basados en la teoría de la evolución de Darwin. Enfrentar un problema con algoritmos genéticos implica buscar una solución mediante mecanismos de selección, cruce y mutación similares a los empleados en la naturaleza, de manera que sobrevivan los más aptos.

Entre las décadas de los 60s y 70s, John Holland, junto con sus alumnos, los desarrolló con el propósito de extrapolar los mecanismos de adaptación natural a

los sistemas computacionales. En su libro *Adaptación en Sistemas Naturales y Artificiales* (1975) quedan establecidos los principios básicos de estos algoritmos [5].

Las poblaciones evolucionan en el tiempo de acuerdo a los principios de la selección natural y la supervivencia de los más aptos. Emulando este proceso, los algoritmos genéticos son capaces de ir creando soluciones para problemas del mundo real [1]. Operan con una población de *individuos* o *cromosomas*, cada uno de los cuales es una representación de una posible solución a un problema dado y compite con el resto por ver cuál constituye la mejor solución. Los individuos están compuestos por *genes*, que representan los parámetros del problema. Para valorar la aptitud de cada solución se dispone de una función de *fitness*. Por medio de esta función se le otorga a cada individuo un valor, que vendría a ser el equivalente al grado de efectividad de dicho organismo para competir por los recursos. Cuanto mayor sea este valor, mayor será la probabilidad de que el individuo en cuestión sea seleccionado para reproducirse y propagar su material genético.



Ejemplo de cruce de doble punto sobre cadenas binarias [4].

La reproducción sexual o *cruce* y las *mutaciones* son los encargados de introducir diversidad genética. El cruce es el principal operador de estos algoritmos y consiste, normalmente, en el intercambio de material genético entre dos cromosomas [2]. Produce por tanto nuevos individuos, descendientes de los anteriores, que combinan características de los padres. Las mutaciones, al igual

que en la naturaleza, aparecen en los algoritmos con muy baja frecuencia y se encargan de producir variaciones en los genes de forma aleatoria.

Dado un algoritmo genético con un máximo de iteraciones *maxIt*, un número de descendientes por cada iteración *n*, un umbral de cruce *Pc* y un umbral de mutación *Pm*, su esquema general queda como sigue:

```
inicialización
mientras iteración < maxIt hacer
   evaluación
   mientras descendientes < n hacer
         selección
         cruce
         mutación
   fmientras
   reemplazo
fmientras
```

Los métodos implicados son:

- *Inicialización*: se crea una población inicial de individuos de manera aleatoria. Se debe asegurar una cierta diversidad genética para evitar la convergencia prematura [3].

- *Evaluación*: evaluamos los cromosomas de la población con la función de *fitness* y determinamos su aptitud.

- *Selección*: seleccionamos un par de cromosomas de la población para cruzar. La probabilidad de que un cromosoma sea seleccionado es proporcional a su función de evaluación.

- *Cruce*: a partir de los padres seleccionados se generan con probabilidad *Pc* dos descendientes que combinan características de ambos.

- *Mutación*: se mutan los genes de los descendientes con una probabilidad *Pm*.

- *Reemplazo*: reemplazamos toda o parte de la población anterior por la nueva población generada.

1.3. Introducción a la Notación ABC

La notación ABC se diseñó para escribir música en formato de texto. Fue inicialmente creada por Chris Walshaw y presentada a finales de 1991. La gran diferencia de ABC con respecto a otros lenguajes musicales orientados a ordenadores es que también puede ser leído con facilidad por las personas. De acuerdo a *music-notation.info* ABC es "tal vez el formato de notación musical más común en internet" [6].

Dado que se trata de una notación basada en los caracteres ASCII, cualquier editor de texto sirve para editar música en ABC. Existen además varios paquetes de software con diversas facilidades que permiten leer y procesar música escrita en el sistema ABC, la mayoría de los cuales son de libre distribución [7].

Un fichero ".abc" está estructurado en dos secciones: la cabecera, que contiene información sobre diversos aspectos de la melodía que se va a exponer a continuación, y las notas que conforman esta melodía. Es en la cabecera donde quedarán indicados aspectos tales como el índex (*X:*), que es el índice que la identifica para cuando se cuente con varias melodías en un mismo archivo; el título (*T:*); la métrica del compás (*M:*); la duración de las notas musicales por defecto (*L:*) o la clave (*K:*). Permite además dejar indicados otros campos como pueden ser el compositor (*C:*) o el origen (*O:*).

Las notas que contempla el estándar de ABC están representadas en notación musical anglosajona.

C	D	E	F	G	A	B
Do	Re	Mi	Fa	Sol	La	Si

Así, se puede manipular la duración de una determinada nota añadiendo una indicación inmediatamente detrás de ésta: C2 dura el doble de la duración por defecto, mientras que C/2 dura la mitad.

Las mismas letras en minúscula representan las mismas notas en una octava mayor: a, b, c, d, e, f, g.

Se utiliza el símbolo de una comilla simple " ‘ " para subir una octava a la nota que sucede. Por ejemplo, "c' ". A su vez, se utiliza el símbolo de una coma "," para bajar una octava. "D," sería un ejemplo.

Por otra parte, los accidentales (sostenidos, bemoles y becuadros) se representan por los caracteres "^", "_", "=" y estos siempre preceden a las letras. Un ejemplo es "^G,".

Para alterar las duraciones de una nota se utilizan números. Estos deben ser siempre positivos y siempre mayor que cero. Por ejemplo, A2 representaría una blanca y A1/2 representaría una corchea si la longitud por defecto fuese una negra en ambos casos.

Los silencios se representan por la letra "z" y también tienen duración asociada.

Se ve a continuación un ejemplo de un fichero en notación ABC para ilustrar su uso:

```
X: 1
T: Ejemplo
M: 6/8
L: 1/4
K: G
A B c d AB cd
```

Analicemos la cabecera: esta melodía lleva por título "Ejemplo", la duración de las notas por defecto es una negra (1/4), la métrica es 6/8 y está escrita en clave de sol. La primera línea indica la numeración de la pieza. Es la forma que utiliza ABC para identificar una pieza con respecto a otras a la hora de generar sus equivalentes MIDI's. Inmediatamente después de la clave se encuentran las notas.

El conocimiento de la notación ABC concede una amplia ventaja al usuario de MyTone, puesto que la aplicación está diseñada para permitir la posibilidad de editar las piezas de muestra y las piezas generadas por el algoritmo de MyTone.

1.4. Estado del Arte

La música evolutiva es música generada a través de algoritmos evolutivos, un subconjunto de algoritmos computacionales que utilizan la inteligencia artificial. Las redes neuronales y los algoritmos genéticos son buenos ejemplos de IA empleada a la hora de componer música.

El objetivo es conseguir que la música de salida sea lo más estética posible a través de sucesivas aplicaciones de pasos computacionales análogos a la selección natural biológica, recombinación y mutación [8].

NEUROGEN

Neurogen (1991) [9] fue de los primeros software que emprendieron esta idea. Utiliza conocimiento extraído de una serie de fragmentos musicales introducidos por el usuario. Con un algoritmo genético, produce y combina los fragmentos musicales y posteriormente evalúa el valor heurístico de cada muestra con una red neuronal entrenada con los ejemplos “reales” de música.

A la hora de delimitar los objetivos de este sistema, los autores se propusieron generar composiciones armónicas de 4 partes diatónicas y de estilo occidental. Todo esto acompañado de una serie de fragmentos musicales valorados por el usuario. Su objetivo principal es el de producir una pieza coherente y que se asemeje a la encontrada en los himnos tradicionales.

Dentro del sistema las redes neuronales procesan distintas muestras musicales que el usuario considera como buena y mala música para capturar las ideas conceptuales que guardan las mismas; por otra parte el algoritmo genético genera numerosos fragmentos musicales parciales que son evaluados con ayuda de la heurística resultante de las redes neuronales, permitiendo así la evolución de aquellos que más se acerquen a los patrones generados por las mismas.

El uso de redes neuronales se ha probado moderadamente efectivo como elemento evaluador de los algoritmos genéticos que, aunque trabajan en un área limitada, producen una calidad de música que cumple con sus objetivos iniciales.

Sin embargo, y tal y como se afirma en el trabajo, aun caben expansiones con el uso algoritmos genéticos múltiples y una reestructuración de las redes neuronales, lo cual deja un marco amplio para posibles ampliaciones futuras.

Algoritmos genéticos y composición musical asistida por ordenador

En este caso y paralelamente a Neurogen, se desarrolló otro de los trabajos pioneros en el campo de la música evolutiva [10], en el que el autor parte de la idea de generar melodías que formen una estructura de puente enlazando un fragmento musical con otro (*thematic bridging*).

El funcionamiento de este sistema consiste en obtener, dadas dos piezas iniciales origen y destino, una pieza final formada por todas las piezas intermedias, sin que la pieza inicial quede incluida en la melodía generada, haciendo así un tipo de música basado en las variaciones de una melodía a otra.

El sistema dispone de una serie de operaciones de modificación y reordenamiento disponibles que actúan sobre la melodía de entrada y que, si tras haber ejecutado el algoritmo genético no llega a la melodía final o el resultado no es satisfactorio para el usuario, se modifica alguno de los parámetros para llegar a una solución. Se sabe también que en este trabajo es necesaria la asistencia del usuario durante el proceso, ya que los cambios se van generando conforme a sus gustos musicales.

También se puede observar que los objetivos alcanzados son limitados por las operaciones y la función de *fitness*. Por ello, el autor da una idea de hacia dónde se puede dirigir su trabajo en proyectos futuros para mejorar lo que, en definitiva, se puede considerar una aplicación satisfactoria.

GenJam

Los algoritmos genéticos también son una parte clave en la improvisación y el acompañamiento de GenJam (1994) [12], abreviación de GeneticJammer. Se trata de una de las obras más reconocidas de por John A. Biles, compositor especializado en aunar la música y las tecnologías.

Este sistema, usado principalmente para improvisar música jazz, es capaz de generar melodías y de cambiarlas o mejorarlas en tiempo real gracias a la evaluación de un usuario y además de componer, puede interactuar con el mismo interpretando la música de entrada y generando variaciones y versiones parecidas en ritmo, armonía y melodía.

Su diseño está basado en el uso de algoritmos genéticos aunque con dos diferencias significativas. En primer lugar se parte de dos poblaciones iniciales, una con información sobre los compases y la otra con un conjunto de melodías.

La otra diferencia es que GenJam utiliza toda la población para generar su salida, sin que se haya evaluado cual de los fragmentos es mejor [11].

Cabe destacar también que este sistema posee dos grandes desventajas: la duración de las notas generadas son múltiplos de ocho y sólo hay catorce tonos disponibles; desventaja que solventan al permitir una mayor diversidad cromática y rítmica.

Como ampliaciones futuras propuestas por el autor se encuentran la inclusión de redes neuronales al sistema, la implantación de melodías existentes como población inicial, o la utilización de la unión de distintas poblaciones generadas en la sesión de entrenamiento para escoger la mejor.

Brian Eno y la música generativa

Fue el propio Eno el que popularizó este término en 1996, tras trabajar con Pete y Tim Cole, diseñadores del software SSEYO koan, para componer su disco "Generativemusic 1", y con él dio origen a esta vertiente de la música generada por ordenador.

El sistema implementado por los hermanos Cole está desarrollado como una arquitectura llamada SSEYO KoanInteractive Audio Platform (SKIAP), que consta de un motor generativo de música (SKME), un juego de herramientas para el usuario (SSEYO Koan Pro and SSEYO Koan X), un reproductor y un plug-in para navegadores de internet [13].

Brian Eno describió este estilo como un tipo de música que está en constante cambio, y así lo presenta en la cubierta de su disco, en la que comenta que, "la música generativa goza de algunos de los beneficios de sus antecesoras. Como la música en vivo, es siempre diferente. Como la música grabada… puedes oírla cuando y donde quieras" [14].

Posteriormente fueron surgiendo sucesivos compositores, la mayor parte de ellos centrados en la idea de usar un algoritmo automatizado, capaz de generar música que resulte agradable a los usuarios.

Sin embargo, no existen incursiones destacables en este campo en las que sea el usuario quien evalúe el *fitness* de cada pieza, a la que se haya dado una libertad total de variación. Ello implicaría escuchar las muestras una a una y evaluarlas, lo que supone un enorme cuello de botella para la aplicación. Se perdería además el algoritmo genético tal y como lo conocemos, dado que la potencia de estos algoritmos es la generación sucesiva de grandes cantidades de individuos con el objetivo de maximizar una función dada, y en este caso el algoritmo no dispondría de dicha función.

MyTone por tanto se inspira en los algoritmos genéticos para generar música pero no los aplica de principio a fin. No se pretende utilizar un algoritmo genético como método de búsqueda para alcanzar una música objetivo ni ideal, puesto que no existen pautas a seguir para conseguir música a gusto de todos. Se pretende emular los principales pasos del algoritmo (la selección, los cruces y las mutaciones) para generar individuos candidatos, que luego el usuario evaluará.

Es gracias a esta cooperación del usuario que el algoritmo desarrollado genera música radicalmente diferente en función de quién lo use. Esto no ocurre en las demás aplicaciones que utilizan un algoritmo genético puro. La función de *fitness*, en estos casos, pretende valorar y puntuar el contenido musical de los individuos siguiendo unos criterios fijados de idoneidad, subjetivos para el que ha desarrollado el programa. Como se ha explicado antes, esto varía mucho en MyTone dependiendo de quién lo use.

1.5. Objetivos y motivación

La motivación de este proyecto nace de la afición por la música de las participantes y del interés que se tenía por experimentar el uso de un asistente de composición que tenga en cuenta los gustos de cada usuario para generar piezas.

Se busca por tanto la creación genuina de música que se ajuste a las tendencias musicales del usuario. Como bien se sabe, la calidad de la música es algo completamente subjetivo y no hay una manera predefinida ni correcta de valorar la calidad de una pieza.

Este aspecto resulta especialmente interesante a la hora de generar música dado que la aplicación de un conjunto de alteraciones (bien sean consideradas poco armoniosas o estridentes o bien sean valoradas como altamente melodiosas) sobre un conjunto de notas puede derivar en motivos musicales considerablemente variados e interesantes.

El objetivo, por tanto, es componer música con un asistente que tenga en cuenta el criterio del usuario y comprobar que la inteligencia artificial detrás del algoritmo generador puede generar música de fondo suficientemente agradable y compleja para el usuario. Esta música de fondo, como bien se sabe, no pretende ser el foco principal de una audiencia sino un acompañamiento a diversas situaciones [15].

También se pretende verificar la inconveniencia del cuello de botella en el algoritmo generador. Se pretende comprobar si este hecho a la hora de generar música impide una experiencia creativa y satisfactoria durante el proceso de composición en MyTone.

Capítulo 2

Tecnologías Utilizadas

En esta sección se explicará detalladamente el software utilizado, sus características y se hará una breve mención a los motivos sobre la elección de las tecnologías utilizadas.

2.1 Java y Netbeans

MyTone se ha desarrollado principalmente en Java en el entorno de desarrollo NetBeans. NetBeans se eligió como IDE (*Integrated Development Environment*) ya que presenta numerosas facilidades para construir y diseñar interfaces gráficas gracias a su panel de componentes Swing y AWT.

La elección del lenguaje de desarrollo se debe principalmente a la experiencia previa con otras aplicaciones programadas en Java y a la alta portabilidad que ofrece éste al ser multiplataforma.

2.2 Notación musical ABC

Como se ha explicado antes, ABC es la notación de música elegida para el proyecto. Una de las motivaciones para utilizar esta notación es que existe un amplio repertorio de software libre y multiplataforma que trabaja con esta notación como *abc2midi* y *abc2mps*.

Estos dos ejecutables interactúan con ficheros de entrada en formato ABC y generan ficheros de salida de audio (MIDI) y representación (PostScript) de la música respectivamente. Ambos son parte del software libre disponible del proyecto *abcMIDI* [16].

Pueden ser descargados gratuitamente desde el siguiente enlace: *http://abc.sourceforge.net/abcMIDI/*

2.3 TiMidity

Para la reproducción de audio se ha utilizado TiMidity, también software libre. Este programa incorpora un sintetizador que utiliza fuentes de audio configurables y permite la inclusión de efectos de audio como las reverberaciones. Además puede convertir el resultado a un fichero de tipo PCM o WAV [17]. Este software también es multiplataforma y se puede descargar de manera gratuita desde el siguiente enlace: *http://timidity.sourceforge.net/*

2.4 Ghost4J

Esta librería ha sido empleada para poder trabajar con ficheros en formato PostScript, empleados en el proyecto para la visualización de las partituras, desde Java. Ghost4J proporciona una API externa capaz de adaptar al lenguaje Java los comandos existentes en el intérprete Ghostscript. Dispone de dos tipos de API diferentes que interactúan con Ghostscript a más alto o bajo nivel [18]. Su descarga gratuita, así como una mayor información acerca del proyecto, está disponible en el siguiente enlace: *http://www.ghost4j.org/index.html*

2.4.1 Ghostscript

Se trata de un intérprete de los lenguajes PostScript y PDF ampliamente reconocido y disponible en múltiples plataformas. Por otra parte, la API de este intérprete sólo está disponible para el lenguaje C y no está adaptada directamente para el lenguaje Java [18]. Ghostscript está desarrollado de forma libre bajo la licencia GNU Afero GPL y puede ser descargado a través del siguiente enlace en donde además se puede encontrar más información al respecto: *http://ghostscript.com/*

2.5 Prolog

Se ha desarrollado un *parser* en Prolog que traduce las notas de entrada en formato ABC a una representación interna que se explicará más adelante. El traductor se implementó utilizando DCG's (Definite Clause Grammars) debido a las facilidades que ofrecen para hacer gramáticas. A partir de esta codificación, se han generado los ejecutables *stand_alone* para que funcionen tanto en Linux como en Windows.

Capítulo 3

Guía de Uso

3.1 Introducción

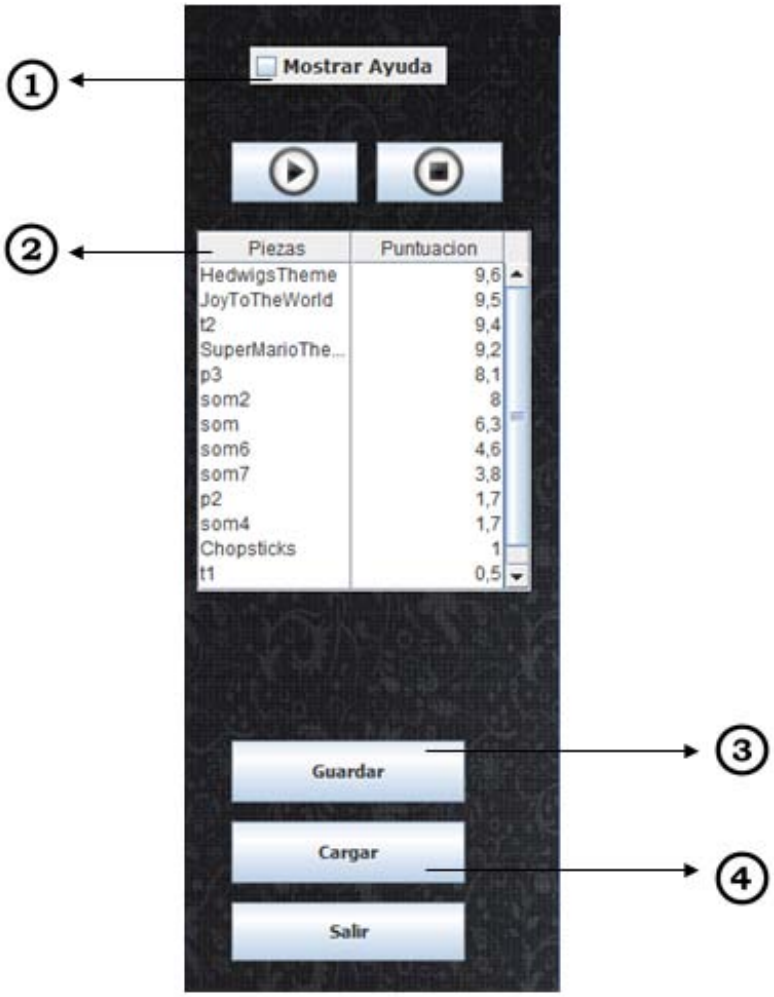
El objetivo de este capítulo es mostrar al usuario las distintas funcionalidades que ofrece la aplicación, así como familiarizarlo con la interfaz. Con este propósito se van a analizar a continuación las diversas secciones que la componen.

Se explicarán las distintas secciones que aparecen en la interfaz: una zona común que siempre está presente en la interfaz (1), la pestaña de edición (2), la pestaña de genéticos (3), la pestaña del arreglador (4) y la pestaña de configuración (5).

3.2 Zona común a la izquierda

Esta sección de la interfaz va a mostrarse siempre al usuario; es decir, es común a las distintas áreas de trabajo de las que dispone MyTone y contiene aquellas funcionalidades y recursos de los que el usuario debe poder disponer en todo momento.

En la siguiente figura se puede observar el aspecto general de la zona común.



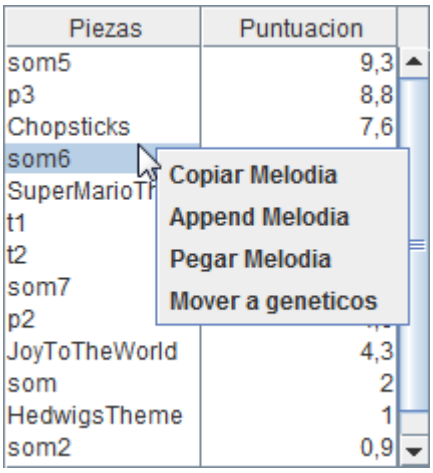
3.2.1 Mostrar Ayuda

Habilitando esta opción se permite que la aplicación oriente al usuario con mensajes de ayuda sobre las funcionalidades de las diferentes áreas de la ventana.

3.2.2 Repertorio de piezas

Cuando arranca la aplicación inicialmente se carga un repertorio de piezas base con las que se procederá a trabajar. Se entenderá por piezas los fragmentos musicales con los que trabaja la aplicación. Estas piezas constan de una única voz. Más adelante se explicará cómo generar piezas de más de una voz.

Para reproducirlas basta con seleccionar una y hacer *click* en el botón de Play, o bien hacer doble *click* sobre la pieza seleccionada. El conjunto de piezas que se muestran en este repertorio se denomina "población".



En esta tabla existe la opción de hacer *click* derecho sobre alguno de los títulos, desplegando un menú con diversas opciones:

a. **Copiar Melodía** copia todo el contenido musical de la pieza seleccionada a una variable para que pueda pegarse en otra pieza.

b. **Concatenar Melodía** pega el contenido musical, almacenado en la variable interna, al final de la pieza seleccionada.

c. **Pegar Melodía** sustituye el contenido musical de la pieza seleccionada por el contenido copiado.

d. **Mover a genéticos** selecciona la pieza elegida como candidata de un grupo selecto de individuos con los que trabajar para el algoritmo de generación. La sección donde esto ocurre se detallará más adelante.

3.2.3 Botones Guardar y Cargar

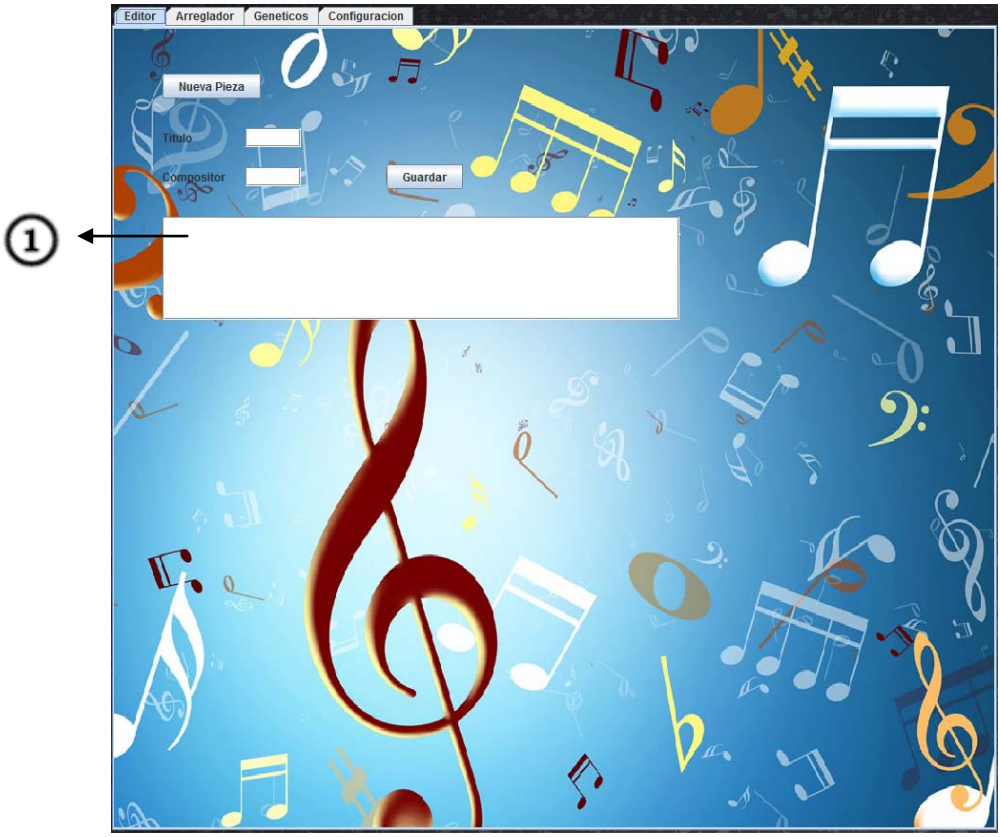
En MyTone se ha habilitado la funcionalidad de cargar y guardar el estado actual de ejecución de la aplicación. El objetivo es permitir que el usuario pueda cerrar su sesión de composición en un momento dado, guardando antes su estado en un fichero de *backup*, y que sea posible la continuación de dicha sesión en cualquier momento, cargando el fichero antes mencionado.

Clickando el botón Guardar se guarda en un fichero el estado de la aplicación: las piezas que componen la población, la composición creada, el perfil de mutaciones y la configuración.

El botón Cargar permite recuperar el estado en el que quedó la aplicación en un momento dado, leyendo del fichero creado al guardar. Restaura las piezas de la población, la composición y la pista de sincronización, el perfil de cruces y mutaciones así como la configuración de los parámetros de la aplicación.

3.3 Pestaña Editor

Esta es la pestaña activa al iniciar MyTone.



Al seleccionar una pieza se puede comprobar que en el área de texto (1) aparece el contenido musical de ésta. En los campos Título y Compositor aparece la información correspondiente asociada a esa misma pieza.

Se puede editar la pieza directamente desde el cuadro de texto, ya sea borrando o añadiendo notas en notación ABC, cambiando el título o cambiando el compositor.

Con el botón **Guardar** se guarda directamente sobre la pieza cualquier cambio realizado. Con el botón **Nueva Pieza** se crea una nueva pieza con el contenido indicado en el área de texto.

3.4 Pestaña Genéticos

Desde esta pestaña vamos a trabajar con la producción de nuevas piezas a partir de otras mediante el algoritmo de generación.

A continuación se detallan los distintos componentes:

3.4.1 Seleccionadas

En esta sección se guardan todas las piezas con las que se trabajará en el algoritmo de MyTone. Si la opción **Toda la población** está seleccionada, se obviará el contenido de esta sección y se aplicará el algoritmo de generación sobre toda la población.

3.4.2 Tabla de Hijos con sus puntuaciones

En esta tabla aparecen los hijos creados a partir del algoritmo. Para evaluar cada pieza bastará con escucharla y asignarle una puntuación comprendida entre el 0 y el 10. Para asociar una puntuación a un hijo se hace doble *click* sobre la columna derecha de la pieza que estamos evaluando ("*hijoX*") y se asigna el valor.

3.4.3 Botón Inspiración

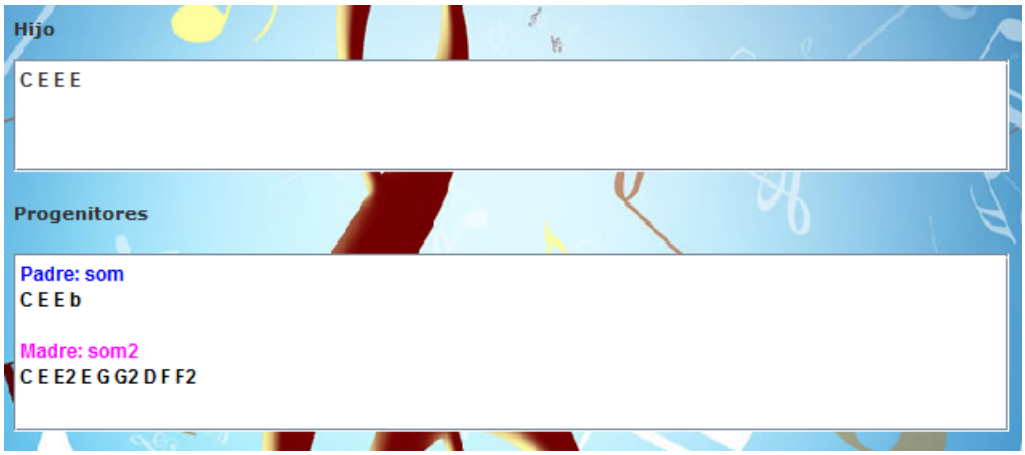
Este botón inicia el algoritmo de generación de piezas. Se eligen dos piezas de las seleccionadas para trabajar con ellas en el algoritmo y aplicándoles el cruce y las mutaciones indicadas por el usuario se generan las nuevas piezas llamadas "hijos". Cuando el algoritmo acabe se mostrarán los hijos generados en su tabla correspondiente como ya se ha explicado arriba.

3.4.4 Botón A Piezas

Pulsando este botón se vacía el contenido de la tabla de hijos y lo vierte a la población actual (tabla de la zona común de la izquierda). A partir de ese momento, el usuario puede empezar a trabajar con los hijos generados. Al pasar a la población, las piezas se reordenan en función de la puntuación asignada de mayor a menor.

3.4.5 Paneles de Hijo y Progenitores

Permite visualizar el contenido del hijo a la vez que se compara y se ilustra qué progenitores tiene. Se debe recordar que el hijo se forma cruzando dos progenitores, y posteriormente aplicando las mutaciones.



3.5 Pestaña Arreglador

En esta pestaña se permite al usuario ordenar las piezas de la población, ya sean piezas base, creadas por el algoritmo o editadas a partir de otras, de modo que pueda ir construyendo su propia composición.

3.5.1 Pista de Sincronización

En esta sección se da la posibilidad de crear puntos de sincronización para la composición. La función de estos puntos es, precisamente, permitir sincronizar piezas pertenecientes a distintas voces. Los puntos aparecen representados en el panel superior por líneas verticales a la altura a la que se sitúen. Al abrir la aplicación, esta pista ya contiene información por defecto.

Veamos ahora cómo introducir nuestros propios puntos. Hay tres tipos de elementos aceptados en esta pista:

- Barras de compás '**|**'

  Estas barras son las encargadas de marcar los puntos de sincronización. Por cada barra que se deje indicada, se calculará la duración de los elementos anteriores a ésta y se creará un punto de sincronización cuyo valor será la suma de esos elementos. Este valor va a marcar a qué "altura" de la duración de la composición se puede sincronizar.

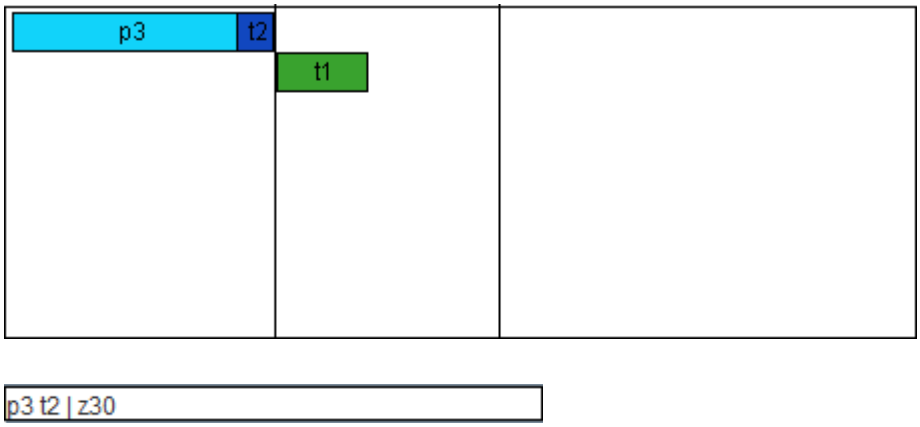
- Elementos de pista

  Los elementos de pista tienen el formato "*z(/)(Num)*", donde el símbolo de división y el *Num* son opcionales. Este formato es semejante a los silencios que maneja la notación ABC, pero es importante que no se confundan: dado que se utilizan para calcular una duración, *Num* representa un número cualquiera dentro del rango de los enteros. La duración de "z" por defecto es la de una corchea (1/8).

- Títulos de piezas

  Títulos de piezas de la población actual también son aceptados. Si el usuario desea añadir un punto de sincronización justo al terminar una determinada pieza en una voz en concreto, esta opción le evita tener que rellenar con Elementos de pista e ir buscando más a ciegas el valor que precisa.

Veamos ahora un pequeño ejemplo:



Primero se han introducido dos títulos que corresponden a piezas de la población seguidas de una barra de compás. El valor del punto de sincronización correspondiente es la suma de las duraciones de ambas piezas. Para mostrarlo gráficamente, se expone una voz formada por estas dos mismas piezas y se puede comprobar que la línea que marca el punto de sincronización se encuentra justo al final de sus representaciones.

Lo siguiente que aparece es un elemento de pista: “z30”. Aunque no se deje indicada ninguna barra de compás detrás, al finalizar el contenido de la pista siempre se añade un punto de sincronización, por lo que ésta sería redundante. El valor del nuevo punto es el equivalente a la suma de las duraciones de las dos piezas del principio y del elemento de pista. La duración de dicho elemento se obtiene multiplicando la duración de una corchea (1/8) por 30 en este caso.

3.5.2 Tabla Arreglador

La tabla consta de 8 filas que se corresponden con 8 voces. En la columna de la izquierda se indican los títulos de las piezas y en la de la derecha el instrumento con el que se desea que suene cada voz.

Para empezar a rellenar la columna de la izquierda se debe seleccionar un título de los disponibles en la población y escribirlo en una de las filas. Es posible añadir más títulos a la misma voz escribiendo más títulos en la misma

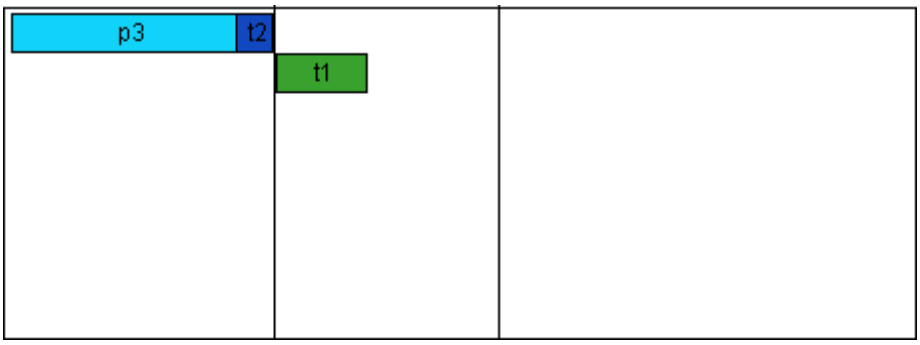
fila y separándolos siempre por espacios. Para indicar que se desea que una pieza en concreto empiece a sonar a partir del siguiente punto de sincronización, se coloca el carácter “>” delante del título de dicha pieza, dejando, como siempre, espacios entre medias. Es posible añadir silencios como si de un título más se tratase, siguiendo el formato “*z(/)(Num)*”, donde el símbolo de división y el *Num* son opcionales.

Después de cada cambio realizado, pulsar *Enter* para actualizar los datos.

En el siguiente apartado se explica con mayor profundidad el funcionamiento de estos caracteres y su visualización en el panel superior con un ejemplo.

3.5.3 Panel Superior

El panel superior muestra de forma gráfica el estado de la composición. Las piezas que forman parte de ésta aparecen representadas por figuras rectangulares con un color fijo y una longitud proporcional a la duración de la pieza. Los puntos de sincronización indicados en la pista anterior quedan representados por líneas verticales de color negro a una distancia también proporcional a su valor.

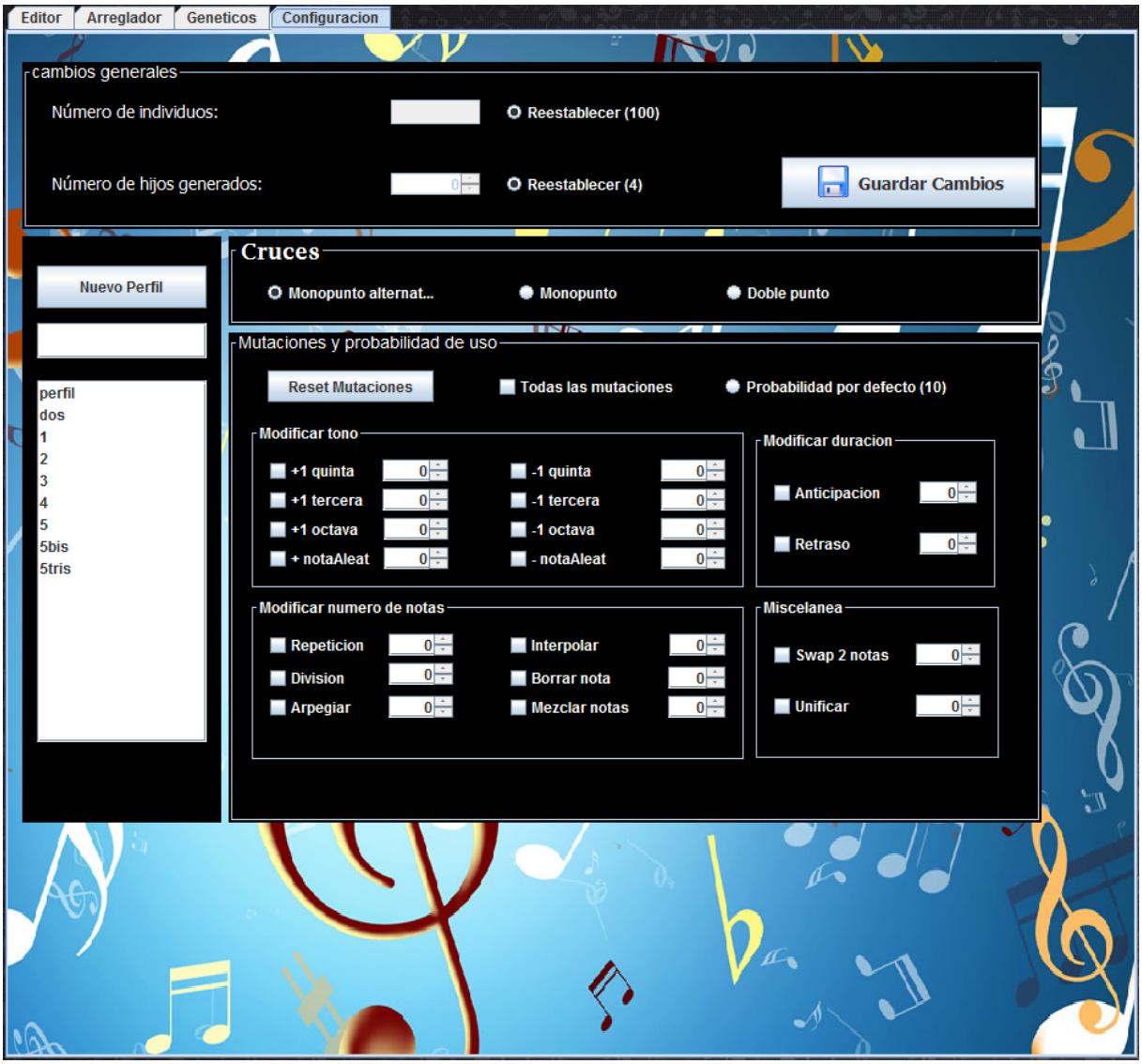


3.5.4 Botón partitura

Este botón genera la partitura de la composición actual. Para ello se realiza el proceso de generación de un fichero ABC a partir de las notas explicado anterior para generar la partitura representativa.

3.6 Pestaña Configuración

Desde esta pestaña el usuario puede modificar algunos parámetros de la aplicación y del algoritmo de generación a su criterio.



3.6.1 Perfiles

Existen una serie de perfiles de mutaciones predefinidos a disposición del usuario. La función de estos perfiles es facilitar la experiencia con MyTone dándole al usuario una configuración de mutaciones ya definida en caso de que no desee especificarla él mismo.

Se ofrece además la posibilidad de crear nuevos perfiles con la configuración indicada al hacer *click* en **Nuevo Perfil**. Los perfiles se almacenan en un

fichero de texto y se cargan al iniciar la aplicación. Para borrar uno, basta con hacer *click* derecho sobre él y seleccionar la opción de **Borrar Perfil**.

3.6.2 Cruces

La sección de configuración de Cruces permite seleccionar el tipo de cruce con el que trabajará el algoritmo de generación de piezas.

- *Monopunto*: las cadenas que conforman las melodías de las piezas padre se intercambian en los hijos a partir de un cierto punto.

- *Monopunto Alternativo*: opera del mismo modo que el cruce Monopunto, pero sólo intercambia las cadenas hasta llegar a la duración original de la pieza de menor longitud.

- *Doble Punto*: los hijos intercambian subcadenas de sus melodías que no tienen por qué ser de la misma longitud.

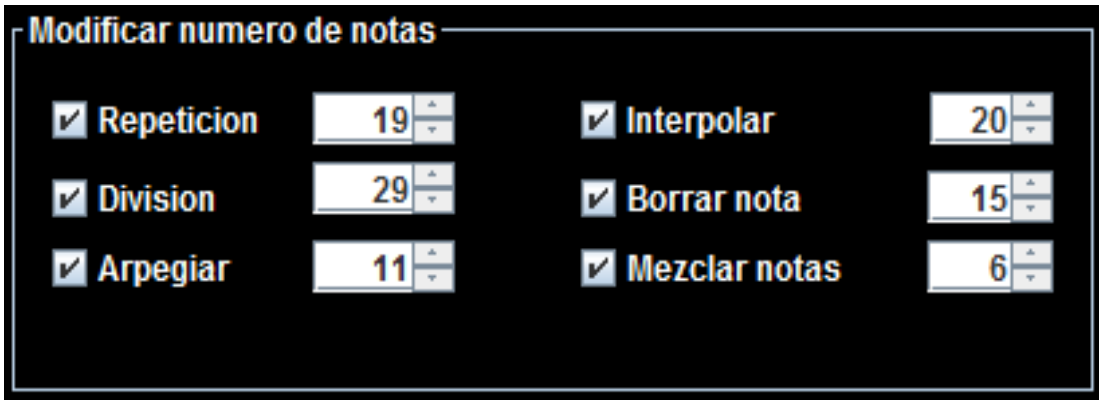
El funcionamiento de estos cruces se explica más a fondo en la sección 5.2.2.

3.6.3 Mutaciones

Aquí se modifican los tipos de mutaciones y con qué probabilidad se darán en el algoritmo de generación. Para indicar que se desea activar una mutación hay que seleccionar su casilla correspondiente. A la derecha de ésta se indica con qué probabilidad debe darse. Si la probabilidad es 0, la mutación no se dará nunca y el resultado sería equivalente a no haber seleccionado la mutación.

El panel nos muestra las mutaciones ordenadas por temática: aquellas que modifican el tono de una nota, las que modifican la duración de la melodía, etc. Para cada uno de estos grupos, la suma máxima de las probabilidades de las mutaciones activadas es 100.

En el ejemplo se ilustra la sección de mutaciones que alteran el número de notas.



3.7 Requisitos de Instalación

MyTone es un software multiplataforma con soporte para Windows y Linux. Dado que utiliza programas externos ya existentes para su funcionamiento, en esta sección se procederá a describir detalladamente el proceso a seguir en la instalación de dichos programas.

Cabe indicar que el programa está disponible en versiones, ejecutables y de código abierto para su posible continuación, para Windows, así como de una versión completa para la plataforma Linux. Partiremos de que para cualquiera de estas versiones es necesario tener instalado el intérprete GhostScript (*http://www.ghostscript.com*).

3.7.1 Versión de código fuente para Windows

En esta versión se provee al usuario de las últimas versiones de estos programas (*abc2midi*, *abcm2ps*, *Timidity*) con lo que no sería necesario ningún tipo de instalación ni descarga.

3.7.2 Versión de código fuente para Linux

En la versión para Linux será necesario descargarse *Timidity* desde el gestor de aplicaciones o mediante línea de comandos. El resto de programas vendrán incluidos en la versión.

Capítulo 4

Arquitectura

4.1 Representación interna de las notas

Para implementar el algoritmo generador de música y poder codificar las mutaciones de manera sencilla y cómoda, la representación que tengan las notas leídas desde el fichero ABC juega un papel clave en el desarrollo de la aplicación.

Es importante remarcar que una nota en ABC se puede representar de más de una manera.

A continuación se ilustra un ejemplo sencillo:

Ambas notas representan el mismo sonido debido a que **c** es de una octava mayor que **C**. Por tanto, con ayuda del símbolo "**,**" se puede bajar una octava a la nota c, consiguiendo así la misma tonalidad.

Por este motivo es conveniente elegir una representación sencilla e unívoca, libre de ambigüedades.

La representación que se ha implementado en MyTone para una nota se resume en esta idea: una nota tiene asociado un tono único, representado por un entero, y una duración.

4.1.1 Nota básica

Una nota básica en ABC consiste en una letra. Si esta tiene una duración diferente de la longitud por defecto, viene seguida de un número. La duración no requiere cálculo alguno puesto que se obtiene de manera inmediata y el tono se deduce siguiendo el esquema a continuación.

	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
MIDI	60	61	62	63	64	65	66	67	68	69	70	71

De este modo, la nota C2 se representaría como el par (60,2). Internamente se estaría hablando de la clase *UnidadMusical*, con dos atributos: el tono y la duración.

4.1.2 Nota de otra octava o con accidentales

Subir o bajar una octava, y alterar una nota mediante un bemol, un becuadro o un sostenido son las modificaciones más frecuentes en una nota. Calcular el tono MIDI en estos casos partiendo de la figura anterior se convierte en una tarea prácticamente trivial.

Los cálculos implicados son:

- Subir o bajar una octava → tono +/- 12
- Bemol → tono – 1
- Sostenido → tono + 1
- Becuadro→ tono inalterado

A continuación se explican los cálculos con un ejemplo sencillo:

Sea la nota " ^G, ". Se conoce el MIDI de la nota "G": 67. Teniendo en cuenta una octava menos (-12) y el sostenido (+1), se obtiene como valor de MIDI el 56.

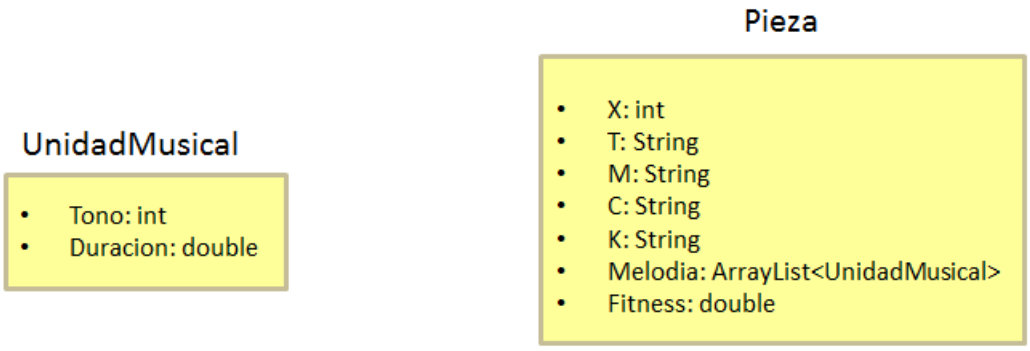
4.2 Representación interna de las piezas

Las piezas en notación ABC contienen una gran variedad de información asociada. La información puede ser referida al título de la pieza, al nombre del compositor, a la longitud de la unidad por defecto, a la métrica de la pieza, o a la clave.

La melodía de las piezas se representa como una lista de objetos de la clase *UnidadMusical* mencionada en el punto anterior.

Por otra parte también se debe almacenar la información referente al *fitness* o puntuación definida por el usuario. Todos los datos mencionados tienen un campo reservado en la clase *Pieza*.

A continuación se ilustran las clases *Pieza* y *UnidadMusical* para facilitar la comprensión de estos conceptos.



4.3 Parser

Para poder trabajar con piezas musicales en ABC se necesita un módulo que haga los roles de un traductor, pasando la notación ABC a la representación interna elegida, antes comentada.

Este componente resulta especialmente útil debido a su reutilización a lo largo de la ejecución del programa. Es decir, no sólo se utiliza para cargar las piezas iniciales en la interfaz sino también para cargar un estado anterior de alguna

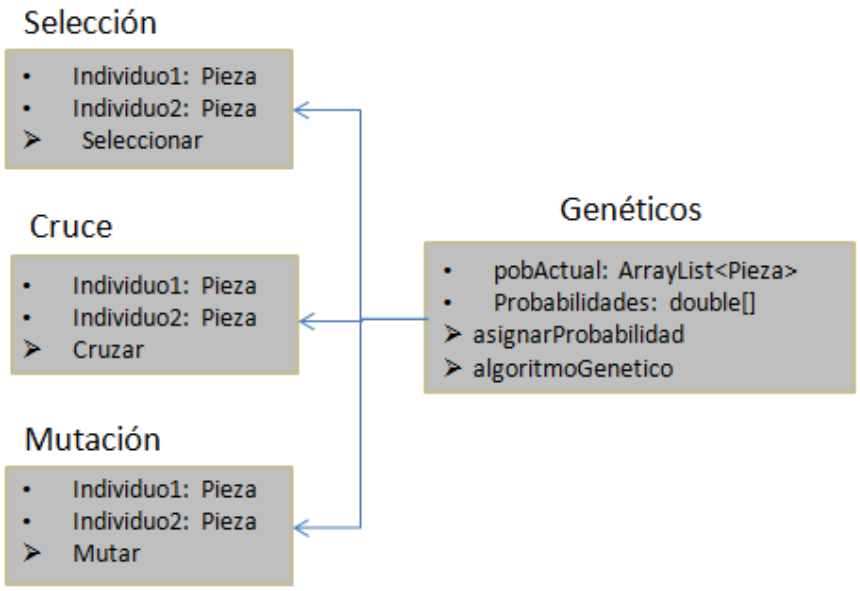
ejecución y para verificar las modificaciones que realice el usuario sobre el contenido de las piezas.

4.4 Algoritmo de Generación

Para desarrollar el algoritmo generador de música se requiere de un módulo que implemente funcionalidades semejantes a las de un algoritmo genético.

Se ha diseñado un módulo que engloba las clases correspondientes a la emulación de las fases de un algoritmo genético. En este módulo se tiene una clase *Genéticos*, una clase *Selección*, una clase *Cruce*, y una clase *Mutación*.

Por otra parte, mencionar que la clase *Genéticos* es una clase de tipo *Singleton* puesto que en toda la ejecución del programa se tiene una única instancia de este objeto.



4.5 Comunicación entre Módulos

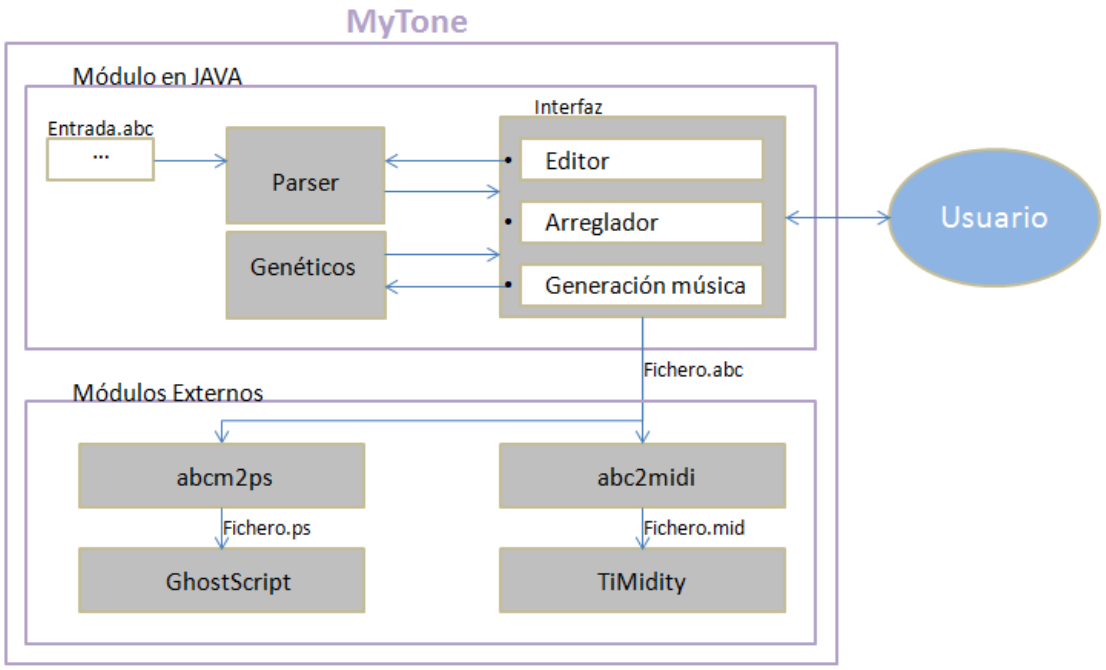
Hacer que funcione MyTone en conjunto requiere un diseño cohesivo y claro que permita que los distintos módulos detallados anteriormente interactúen de forma correcta entre ellos.

La aplicación requiere de una carga de piezas iniciales que conforman la población actual. Estas piezas están almacenadas en un fichero de texto de tipo ABC ("*entrada.abc*") que es leído, y reconocido por el *parser* y posteriormente traducido a la representación interna de una pieza musical explicada en los puntos anteriores.

Si se desea reproducir algo en un momento dado, MyTone genera el fichero ABC correspondiente ("*fichero.abc*") y ejecuta el programa **abc2midi,** que le proporciona el archivo de audio ("*fichero.mid*"). Teniendo este archivo generado, MyTone lanza el programa **TiMidity** para reproducirlo.

Por otra parte, los procesos de guardar y cargar se tramitan a través de un fichero ("*Estado.txt*"). Éste se genera cuando se guarda el estado de la aplicación en algún momento de la ejecución y permite restaurar una sesión de composición previa haciendo uso de la opción **Cargar**.

Se ilustra un ejemplo para resumir cómo se interconectan los distintos módulos en MyTone.


MyTone
Módulo en JAVA
Entrada.abc
...
Parser
Genéticos
Interfaz
Editor
Arreglador
Generación música
Usuario
Fichero.abc
Módulos Externos
abcm2ps
Fichero.ps
GhostScript
abc2midi
Fichero.mid
TiMidity

Capítulo 5

Desarrollo

5.1. Parser

El módulo que transforma la entrada en formato ABC a la representación interna de las notas, comúnmente referido como *Parser*, se ha implementado en Prolog utilizando DCG's (*Definite Clause Grammars*). Además de reconocer las notas musicales de los ficheros de entrada, se ha reutilizado su código para otras funciones como comprobar la corrección en la edición de piezas y la carga de estados guardados durante la ejecución de la aplicación.

La gramática básica reconocedora de los elementos musicales tiene el siguiente aspecto; nótese que el símbolo | se ha usado para indicar otra regla proveniente de la misma producción.

```
composición ->sinesp, nota, notas | "" , tuplet, notas

sinesp ->esp,! , sinesp| "" .

esp -> " " | "\n" .

sep -> "|" | "||" |esp| "( " | ")" | "" .

notas ->sinesp, nota,!, notas | sinesp, tuplet,!, notas| "|" ,! | "||" ,! | "" .

nota ->multiple | sencilla | silencio

sencilla -> accidental, letra, octava, duracion

accidental -> ^ | _ | = | ""

octava -> " '" octava | "," octava | ""

duracion -> entero "/" entero | "/" entero | "/" | entero | entero "/" | ""

letra ->letraMay|letraMin

letraMay -> "A" | "B" | " C" | "D" | " E" | "F" | " G"
```

```
letraMin -> “a” | “b” | “c” | “d” | “e” | “f” | “g”

multiple ->slur

tuplet -> “( “ entero sencillas

slur-> sencilla “- “ sencilla

sencillas -> sencilla restosencillas

restosencillas ->sinesp, sencilla, restosencillas | “”

silencio ->letraSil duración

letraSil -> “z” | “Z”
```

A continuación se enumeran los elementos reconocibles por esta gramática:

a) Notas de cualquier octava
b) Sostenidos, bemoles y becuadros
c) Notas de cualquier duración
d) Ligaduras
e) Tuplas: dosillos, tresillos, cuatrosillos, cinquillos, seisillos y octillos
f) Silencios

Para generar las tuplas representativas de las notas se ha procedido a *decorar* la gramática. Este proceso consiste en añadir parámetros a las producciones para posteriormente operar con ellos.

A continuación se muestra como ejemplo la producción de una sencilla decorada.

```
sencilla([(N,D)]) --> accidental(A), letra(L), octava(O), duracion(D),
{
   N is L+A+O
}.
```

Como se puede observar, el valor del tono de una nota es el resultado de la suma de su tono base (indicado por L), de su octava (indicada por O, con valores múltiplos de 12 o -12), y del accidental (indicado por A con valores posibles -1, 0 y +1).

Tras haber procesado todas las notas en notación ABC se genera una lista con listas de tuplas. Cada lista interna es una lista unitaria con la tupla representativa de la nota.

Para comunicar este módulo con el módulo implementado en Java se ha procedido a utilizar ficheros de entrada y salida.

El *parser* tiene como entrada un archivo de texto plano con notas musicales en notación ABC, procedente del módulo en Java. Tras reconocer con éxito la entrada se genera una lista de pares en otro fichero de texto plano. Este fichero de salida es el que se lee desde la plataforma en Java y mediante él se construyen las piezas musicales con las cuales se trabaja durante la ejecución del programa.

El tratamiento de errores en la sintaxis del fichero de entrada se lleva a cabo dejando de reconocer la entrada en el momento de la detección de una incoherencia y mostrando un mensaje de error. En el fichero de salida se guardan las tuplas que se han procesado hasta el momento del error seguido por un "-" para indicar que la lista que sucede al símbolo contiene errores.

Por ejemplo, si se tuviese en la entrada la cadena "e d2 j a b c", siendo el carácter "j" un error, en la salida se tendría que:

```
[[ (76,1)],[ (74,2)]]-[j,a,b,c]
```

Se puede comprobar que el símbolo "-" separa la lista de tuplas reconocidas de la lista de caracteres no reconocidos.

El símbolo "+", por otra parte, se usa para indicar que se ha reconocido con éxito la cadena entera de notas musicales, y este se encontraría al final de la lista de tuplas.

5.2. Algoritmo de generación

El algoritmo de generación de piezas a partir de otras ya creadas está enteramente inspirado en los algoritmos genéticos, por lo que su estructura es muy similar. A continuación se exponen los fundamentos y detalles de implementación de los distintos apartados que lo conforman.

5.2.1 Selección

Como ya se mencionó anteriormente, en nuestro caso es el usuario el encargado de evaluar el *fitness*. Tendrán más probabilidad de ser seleccionadas aquellas piezas que maximicen esta función; es decir, las que más hayan gustado al usuario.

MyTone puede operar sobre todos los individuos de la población disponible seleccionando la opción "Toda la población" de la pestaña Genéticos, pero ofrece también la posibilidad de que sea el usuario quien determine qué piezas formarán parte del algoritmo y tendrán opción a generar otras nuevas; para ello sólo tiene que mandar los individuos deseados a la tabla "Seleccionadas", también en la pestaña Genéticos.

Una vez que ya se dispone de los individuos sobre los que se va a trabajar, puede comenzar el algoritmo de selección. Se detallarán los pasos a continuación:

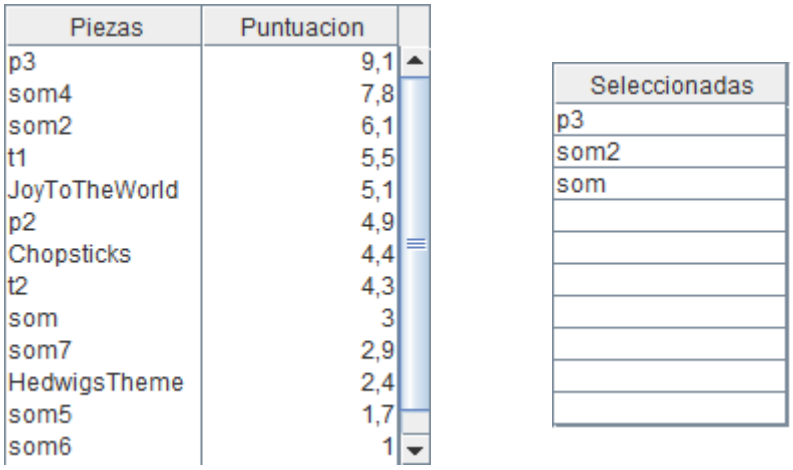
1) El primer paso consiste en recorrer uno por uno los individuos afectados e ir calculando la suma de sus *fitness*, dado que ese valor será necesario para continuar.

2) Una vez que disponemos de la suma de los *fitness*, se pasa a calcular la puntuación de cada pieza relativa a la población. Dicha puntuación viene dada por

$$puntuacion = \frac{fitness}{sumaTotalFitness}$$

Este valor representa en cierta forma el "peso" o la "importancia" de un individuo con respecto a la sociedad; en el caso del algoritmo sería el "peso" de la pieza dada con respecto al conjunto de las seleccionadas para trabajar con ellas. Aquellas con un *fitness* mayor tendrán a su vez una puntuación mayor, y para las que menos hayan gustado su puntuación será menor.

Veamos un ejemplo muy sencillo: se suponen los siguientes individuos en la población y la siguiente tabla de seleccionadas por el usuario:



Piezas	Puntuacion
p3	9,1
som4	7,8
som2	6,1
t1	5,5
JoyToTheWorld	5,1
p2	4,9
Chopsticks	4,4
t2	4,3
som	3
som7	2,9
HedwigsTheme	2,4
som5	1,7
som6	1

Seleccionadas
p3
som2
som

La suma total de los *fitness* de las seleccionadas será 18.2. Las puntuaciones de cada individuo serán entonces

$$p3.puntuacion = \frac{p3.fitness}{sumaFitness} = \frac{9.1}{18.2} = 0.5$$

$$som2.puntuacion = \frac{som2.fitness}{sumaFitness} = \frac{6.1}{18.2} = 0.335$$

$$som.puntuacion = \frac{som.fitness}{sumaFitness} = \frac{3}{18.2} = 0.165$$

3) Con las puntuaciones ya calculadas, lo único que falta por obtener son las puntuaciones acumuladas. Éstas, como ya indica su nombre, representan la suma o acumulación de las puntuaciones de los individuos anteriores y la del individuo en cuestión. Sirven para organizar las puntuaciones de los cromosomas de la población en el

rango entre 0 y 1. La suma de las puntuaciones de los individuos anteriores (*sumaP*) se inicializa a cero.

$$sumaP = 0$$

$$p3.puntuacionAcumulada = sumaP + p3.puntuacion = 0 + 0.5 = 0.5$$

$$sumaP = sumaP + p3.puntuacion = 0 + 0.5 = 0.5$$

$$som2.puntuacionAcumulada = sumaP + som2.puntuacion = 0.5 + 0.335 = 0.835$$

$$sumaP = sumaP + som2.puntuacion = 0.5 + 0.335 = 0.835$$

$$som.puntuacionAcumulada = sumaP + som.puntuacion = 0.835 + 0.165 = 1$$

$$sumaP = sumaP + som.puntuacion = 0.835 + 0.165 = 1$$

4) Ya tenemos todos los atributos necesarios calculados. El siguiente paso es seleccionar un individuo con probabilidad proporcional al peso de su *fitness* en la población. Para ello se genera un número aleatorio entre 0 y 1. El primer individuo cuya puntuación acumulada supere ese valor, será seleccionado para cruzarse.

   Por ejemplo, si el número generado es 0.79, el individuo seleccionado será som2, con 0.835 de puntuación acumulada. Si el valor es 0.13, el seleccionado será p3. Vemos entonces que aquellos cuya puntuación abarca un mayor rango son los que más probabilidad tienen de ser seleccionados; es decir, que el algoritmo favorece a aquellos que tengan mayor *fitness*.

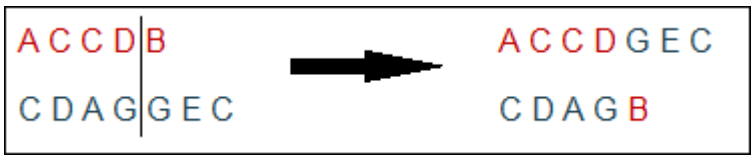
Cada vez que se ejecuta el algoritmo de selección se eligen dos piezas con las cuales se va a trabajar durante el resto de la ejecución del algoritmo de generación.

5.2.2 Cruce

El algoritmo de selección nos devuelve parejas de individuos con las cuales se trabaja en el cruce. Por cada pareja se generan dos hijos que están formados por genes de su padre y de su madre. Todos los cruces desarrollados en MyTone son cruces sexuales; es decir, se necesitan dos progenitores para generar los hijos. El umbral de cruce en la aplicación es 1 de base, lo que implica que las piezas de una pareja siempre se van a cruzar. Esto se debe básicamente a que la aplicación busca generar nueva música que el usuario pueda valorar y con la que trabajar. Dada la función de MyTone, no tiene demasiado sentido devolver al usuario la misma pieza que ya ha introducido sin haber sido cruzada, puesto que ya puede disponer de esa pieza cuando desee. Por otra parte, el algoritmo genético implementado no sustituye a los padres por los hijos generados, por lo que el usuario puede seguir trabajando con los padres y con las nuevas piezas.

El primer paso es generar dos nuevas piezas, los hijos, con las cuales vamos a trabajar durante el algoritmo. Se almacena para cada uno quiénes son su padre y su madre con la intención de valernos de esa información más tarde. Los siguientes pasos van en función del tipo de cruce seleccionado por el usuario. Lo vemos a continuación:

- *Cruce Monopunto*: el objetivo es intercambiar la información genética a partir de un determinado punto. Por tanto, lo primero que se debe hacer es seleccionar dicho punto. MyTone acepta piezas con melodías de distinta longitud, así que para elegir ese índice debemos asegurarnos de que se encuentre en el rango de la melodía más corta de la pareja.



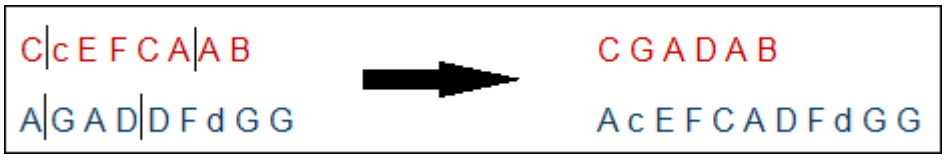
Ejemplo de cruce monopunto.

Determinamos primero cuál es la melodía más corta comparando el número de genes de ambas. Una vez que sabemos esto y que conocemos el número de genes que la conforman, se calcula el *punto de cruce* de cada gen, que viene determinado por la ecuación:

$$\frac{Índice\ gen}{Número\ total\ de\ genes}$$

Se genera un número aleatorio entre 0 y 1. Cruzaremos desde el primer gen cuyo punto de cruce supere ese valor. En la imagen el punto de cruce se sitúa en el índice 5. Los hijos conservan los genes de uno de los padres hasta ese punto, a partir del cual los intercambian.

- *Cruce de Doble Punto*: en este caso vamos a intercambiar el material genético comprendido entre dos puntos de una de las melodías por el comprendido entre otros dos puntos de la otra melodía. Tomamos el primer hijo y seleccionamos dos índices *i*, *j* tal que *i* < *j*. Procedemos de la misma forma con el segundo hijo, obteniendo los índices *p* y *q* tal que *p* < *q*. A continuación intercambiamos los genes comprendidos entre *i* y *j* por los comprendidos entre *p* y *q*.



Ejemplo de cruce de doble punto.

- *Cruce Monopunto Alternativo*: al igual que con el cruce monopunto, la información genética se intercambia a partir de un determinado punto. La manera de proceder para seleccionar dicho

punto es la misma que ya se mencionó en esa sección. La diferencia radica en el intercambio, que será el mismo salvo porque sólo se intercambiarán los genes desde el punto de cruce seleccionado hasta la longitud de la melodía más corta. Veamos un ejemplo que ilustra la diferencia entre ambos:

Cruce monopunto alternativo vs monopunto.

5.2.3 Mutación

MyTone dispone de una amplia variedad de mutaciones, inspiradas en [19,20], que resultan visibles desde la pestaña de Configuración. El usuario puede seleccionar aquellas con las que desea trabajar y la probabilidad con la que se darán durante el algoritmo. Tanto externa como internamente, las mutaciones están divididas en grupos según su efecto:

- Modificar la duración de una nota [Grupo 1]
- Modificar el tono de una nota [Grupo 2]
- Modificar el número de notas de la melodía [Grupo 3]
- Intercambiar dos notas [Grupo 4]
- Uniformizar una nota con respecto a las que la rodean [Grupo 5]

De ahora en adelante nos referiremos a ellos por el número indicado a la derecha de cada uno. Dentro de cada grupo las mutaciones son mutuamente excluyentes y la suma máxima de las probabilidades asignadas a éstas es 100. Por ejemplo, si el usuario activase las mutaciones “subir una octava” y

“subir una quinta”, cada gen podría verse afectado como mucho por una de ellas; sin embargo, de estar activadas “subir una octava” y “swap de notas”, el gen podría verse afectado por ambas, ya que no pertenecen al mismo grupo. A continuación veremos esto con más detenimiento.

La mutación opera sobre las piezas generadas como resultado del cruce, recorriendo los genes de ambas uno por uno y comprobando si se ven afectados por alguno de los posibles cambios o mutaciones. Durante el proceso se mantienen tres tablas hash, una por cada grupo de los anteriormente mencionados que esté formado por más de una mutación (grupos 1, 2 y 3). Estas tablas contienen claves de 1 a 100, y sus posibles valores van de 0 al número de mutaciones que conforman cada grupo; el 0 implica que no hay mutación, el resto de números indican la clase de mutación por la que se vería afectado el gen.

- *Cómo rellenar una tabla*: se recorren las mutaciones del grupo. Para aquellas que el usuario haya activado, se toma su probabilidad y se introduce en la tabla el número de la mutación para tantas claves como porcentaje tenga la probabilidad.

  Supongamos activadas las mutaciones “subir una octava” con probabilidad del 30%, “bajar una quinta” con probabilidad del 15% y “bajar una octava” con probabilidad del 24%. Todas ellas son mutaciones pertenecientes al grupo 2. Sabiendo que “subir una octava” se corresponde con la mutación número 3, “bajar una quinta” con la 4 y “bajar una octava” con la 6, la tabla hash correspondiente a dicho grupo quedaría entonces:

  Para las claves de 0 a 29, el valor es 3; de 30 a 44, es 4; de 45 a 68, es 6; de 69 en adelante, el valor es 0, que indica que no se aplicará ninguna mutación.

CLAVE	VALOR
0	3
...	3
29	3
30	4
...	4
44	4
45	6
...	6
68	6
69	0
...	0
99	0

- *Cómo usar una tabla*: se genera un número aleatorio entre 0 y 100, éste último excluido. De la tabla tomamos el valor de la clave correspondiente al número obtenido. Ese valor corresponde a la mutación que se le va a aplicar al gen o, si es 0, entonces se deja intacto. De esta forma se logra que las mutaciones sean mutuamente excluyentes dentro de cada grupo, evitando casos de anulación (se sube una octava el gen y luego se le baja) o acumulación excesiva de mutaciones de la misma temática (subirlo una quinta, una tercera y una octava).

Una vez disponemos de las tablas rellenas, recorremos todos los genes de cada individuo obtenido como resultado del cruce. Utilizamos las tablas de la manera indicada para aplicar las mutaciones pertinentes. Para aquellos grupos que no cuenten con tabla porque sólo constan de una mutación, comprobamos directamente si se aplica de la siguiente manera:

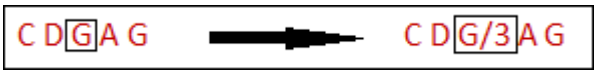
Se genera un número aleatorio entre 0 y 100, éste último excluido, igual que se hace trabajando con las tablas. Si el valor resultante es menor que la probabilidad impuesta por el usuario para la mutación que conforma el grupo, entonces ésta se aplica. De lo contrario, el gen no se ve afectado por ella.

Veamos ahora las distintas mutaciones que forman parte de cada grupo:

- **Grupo 1: Modificar la duración de una nota**

  Como indica el nombre, este grupo va a estar formado por todas aquellas mutaciones que afecten al gen modificando su duración.

  1. Anticipar
     Reduce la duración del gen a un tercio de la original.



  2. Retraso
     Multiplica por tres la duración original del gen en cuestión.

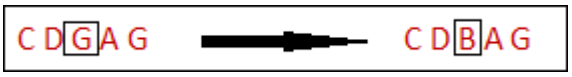
- **Grupo 2: Modificar el tono de una nota de forma específica**

  En este grupo se encuentran aquellas mutaciones que modifican la nota en un número específico de semitonos. Se tratan en general de intervalos consonantes, dado que se intenta no introducir demasiada disonancia.

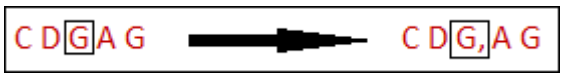
MUTACIÓN	ACCIÓN	NÚMERO DE SEMITONOS
Subir una tercera	Sumar	4
Subir una quinta	Sumar	7
Subir una octava	Sumar	12
Subir aleatorio	Sumar	Aleatorio
Bajar una tercera	Restar	4
Bajar una quinta	Restar	7
Bajar una octava	Restar	12
Bajar aleatorio	Restar	Aleatorio

Veamos un par de ejemplos:

1. Subir una tercera
   Aumenta el tono de la nota en cuatro semitonos.



2. Bajar una octava
   Reduce el tono de la nota en doce semitonos.

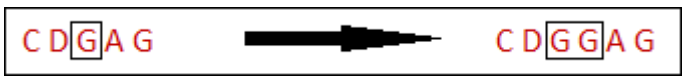


Las mutaciones Subir aleatorio y Bajar aleatorio no se corresponden necesariamente con intervalos consonantes, como su propio nombre da a entender. Por tanto ambas son capaces de introducir disonancia.

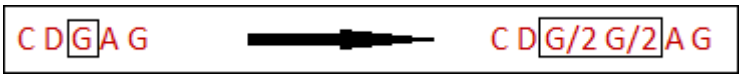
- **Grupo 3: Modificar el número de notas de la melodía**

Según la mutación, se puede añadir una nota al final, entre medias o eliminarla.

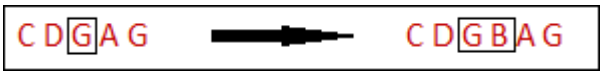
1. Clonar
   Crea una copia del gen y la introduce a continuación de éste.



2. División
   Reduce la duración del gen a la mitad e introduce una copia a continuación con la duración también reducida.

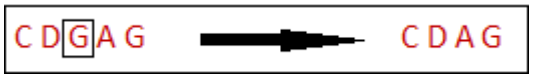


3. Arpegiar
   Se crea una copia del gen en la que se modifica el tono subiéndolo una tercera. La copia se introduce en la melodía a continuación del gen afectado.

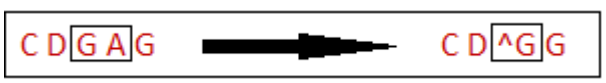


4. Interpolar
   Se introduce un nuevo gen a continuación del afectado cuyo tono va a ser la media de éste y del siguiente gen en la melodía.

5. Eliminar Nota
   Como ya indica el título, esta mutación borra el gen afectado de la melodía.



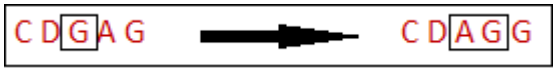
6. Combinar
   Fusiona el gen afectado y su siguiente en uno solo cuyo tono y duración son la media de los tonos y duraciones de ambos genes a los que sustituye.



- **Grupo 4: Intercambiar dos notas**

  Este grupo consta de una sola mutación, así que es uno de los dos que no necesitan tabla para administrarlos.

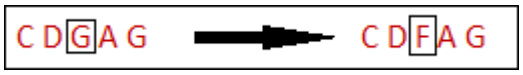
  1. Swap de Notas
     Se intercambian el gen en cuestión y su siguiente.



- **Grupo 5: Uniformizar una nota con respecto a las que la rodean**

  Este grupo también consta de una sola mutación, así que tampoco necesita tabla.

  1. Unificar
     El tono del gen afectado pasará a ser la media de los tonos del gen que lo precede y del anterior. La duración queda intacta.



5.3. Perfiles

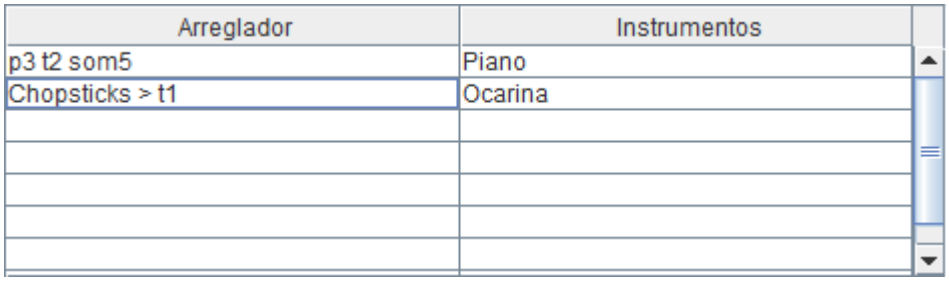
MyTone dispone de unos perfiles de mutaciones predefinidos a disposición del usuario. Al seleccionar uno de estos perfiles se carga la configuración de mutaciones asociada, donde vienen activadas unas ciertas mutaciones cada una con una cierta probabilidad de darse en el algoritmo.

Los perfiles se cargan al iniciar la aplicación leyéndolos de un fichero en el que están almacenados. Para crear un nuevo perfil sólo se necesita darle un nombre que no coincida con el de ningún otro perfil. La configuración de mutaciones que se encuentre activa se guardará entonces en el fichero de perfiles con su nombre asociado.

Para eliminar un perfil –*click* derecho, Borrar Perfil- se busca primero su nombre en el fichero de perfiles. Una vez encontrado, se borra éste y la información de su configuración.

5.4. Arreglador

Este componente es clave para la composición musical en MyTone. Es sobre él que el usuario especifica qué piezas desea incluir, en qué orden, voz y con qué instrumentos deben sonar.



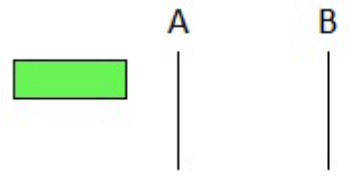
Al actualizar la tabla se comprueba la validez de lo especificado. Se utiliza una matriz para almacenar los títulos de los elementos reconocidos en su fila y columna y otra para almacenar las duraciones. Lo primero es recorrer la primera columna de cada una de las ocho filas de la tabla.

Para cada uno de estos campos se comprueba uno por uno sus elementos para verificar que son aceptados. Cada vez que se identifica un elemento, que está separado del siguiente por, como mínimo, un espacio, se trata de encontrar un título de la población de piezas actual que corresponda con dicho elemento. De encontrarlo, se introduce su título y duración en las matrices correspondientes. Si

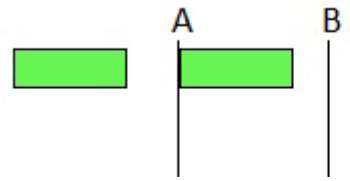
no se encuentra en la población, se comprueba si cumple el formato de un silencio. De ser así, se calcula su duración y se introduce en la matriz de duraciones. En la matriz de títulos se introduce entonces el elemento en sí. Si el elemento no está en la población y tampoco es un silencio, puede tratarse del símbolo de sincronización ">" o de un error. En el segundo caso, el algoritmo almacena los elementos que no ha encontrado y se los muestra posteriormente al usuario. El primer caso es más complejo y lo vemos más a fondo a continuación.

Si el elemento se corresponde con el símbolo ">" implica que se debe asegurar que los elementos que vengan a continuación empiecen a partir del siguiente punto de sincronización. Se calcula entonces la suma de las duraciones de los elementos que lo preceden. Una vez que se dispone de ese dato, recorremos los puntos de sincronización en busca del primero cuyo valor supere dicha suma, asegurando así que el punto de sincronización sea el primero más cercano pero posterior. La estrategia entonces es rellenar con silencios el espacio desde donde nos encontramos hasta dicho punto. En función de la "distancia" o duración a rellenar, se introducen unos silencios de una duración u otra de manera que se ajusten al valor buscado. Finalmente se sustituye el símbolo de sincronización ">" por esos silencios y el algoritmo los evalúa todos como nuevos elementos.

Supongamos "som6":



"som6 > som6":



Al encontrar el símbolo de sincronización, se calcula la suma de las duraciones de los elementos previos, que en este caso se limita a la duración de *som6*. El primer punto de sincronización que supera este valor es A, por lo que los siguientes elementos deben situarse a partir de ese punto. Para ello se rellena la pista con

silencios que no son visibles al usuario hasta alcanzar A. Vemos que la siguiente pista aparece donde debe.

5.5. Pista de sincronización

Este área de texto puede ser manipulada por el usuario en cualquier momento. Al pulsar *Enter* se activa el evento de este componente, responsable de almacenar los puntos de sincronización con sus respectivos valores y, posteriormente, repintar el panel de visualización de composición, del cual hablaremos en la siguiente sección.

El primer paso es obtener la cadena de contenido. Partimos ésta en varias subcadenas dividiéndola por la barra de compás. Para cada subcadena, calculamos la suma de las duraciones de los elementos que la componen.

- Títulos de piezas

  Se busca en la tabla de títulos uno que corresponda con el introducido. De encontrarlo, obtenemos el índice que ocupa en la lista de piezas y a través de éste obtenemos la pieza a la que se hace referencia. Al tener acceso a la pieza podemos conocer su duración.

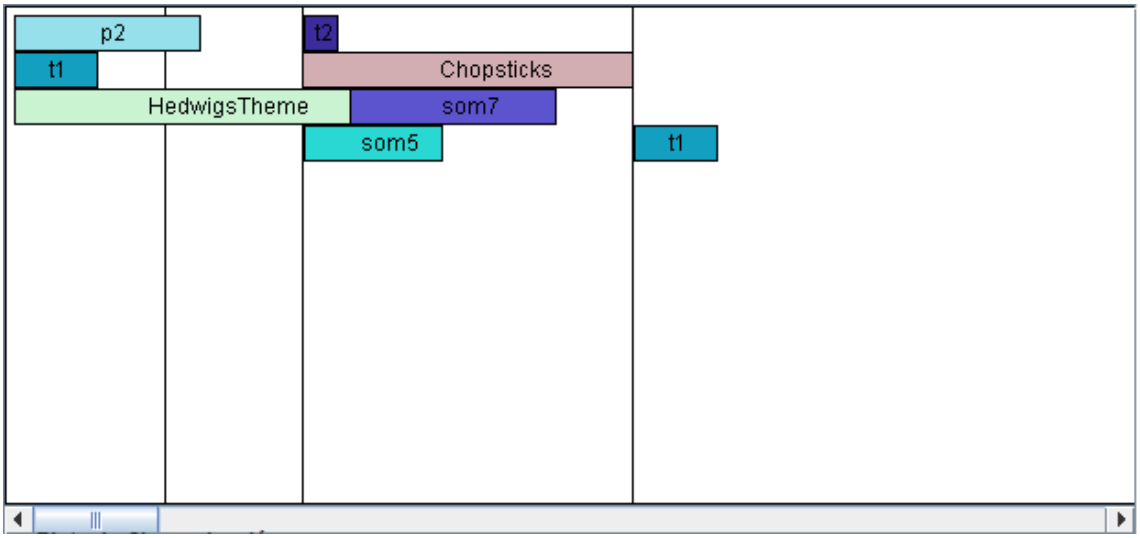
- Elementos de pista

  En caso de no concordar con ningún título de la población de piezas, se comprueba que sea un elemento de pista. Se determina si cumple con el formato adecuado para ello y, de ser así, se calcula la duración multiplicando o dividiendo la duración de la “z”, 0.125, por el número indicado a continuación.

Una vez conocida la suma de las duraciones de los elementos pertenecientes a una subcadena, se crea un nuevo punto de sincronización añadiendo su valor a una lista de ellos. Este valor viene dado por la suma del cálculo de duración para la actual subcadena y la calculada para subcadenas anteriores.

5.6. Visualización de composición

Este panel es el encargado de representar gráficamente el estado de la composición. Vemos ahora cómo funciona internamente.



Como se comentó antes en el apartado del Arreglador, cada vez que se actualizan los datos de la tabla del arreglador, se recorre cada fila, se *parsean* los títulos, silencios o elementos de sincronización y se calcula su duración correspondiente.

Por otra parte, a cada título reconocido en la tabla del arreglador se le asocia un color. Las asociaciones entre los títulos y colores se guardan en una estructura de datos de tipo HashTable, con el título como clave, y el color como valor.

La forma de asignar los colores a los títulos es comprobando si el título se encuentra en el HashTable. Si está, se utiliza el color ya asignado para esa pieza. Si no está, se inserta ese título junto con un color calculado aleatoriamente, comprobando siempre que el color no esté asignado a otro título (de ser así, se repetiría el proceso hasta dar con un color que no esté asignado).

Al rellenar el panel, lo primero que se pinta son los puntos de sincronización. Se recorre la lista en la que se almacenan y para cada uno de ellos se dibuja una línea vertical a una altura proporcional a su valor.

Para dibujar las piezas se recorren las matrices de títulos y de duraciones fila por fila y columna por columna. Por cada título se pinta un rectángulo de ancho proporcional a la duración que le corresponde. Los silencios no se dibujan, pero al

encontrar uno, el punto a partir del cual se sigue dibujando se desplaza a la derecha una distancia proporcional a la duración de este silencio.

5.7. Guardar y Cargar

El estado dela ejecución del programa viene definido por tres aspectos fundamentales: el estado del conjunto de piezas en la población actual, el estado de la sección del arreglador y el estado de las sucesivas opciones configuradas sobre el número de hijos, las mutaciones, los cruces, y los perfiles creados.

5.7.1 Población actual

Guardar la población actual conlleva guardar todas las piezas que se muestran en el área común a la izquierda de la interfaz. Esto quiere decir que se tiene en cuenta el valor del parámetro Número de individuos previamente configurado.

Se empieza guardando el contenido de cada pieza. Primero los valores de los campos de una pieza, posteriormente la melodía y finalmente su *fitness* asociado. Para almacenar la melodía, ésta se convierte a una cadena de notas que siguen el estándar ABC.

Para cargar la población actual se procede de una manera muy similar a la carga de las muestras iniciales presentadas al arrancar MyTone. La principal diferencia entre ambas reside en que en este caso se asigna un valor de *fitness* determinado a cada pieza.

5.7.2 Composición

La composición, según se ha explicado antes, se construye a partir de las melodías de otras piezas. Por tanto, para guardar su estado no basta con guardar su contenido musical; se requiere además el título de cada pieza que conforma dicha composición.

Para ello se recoge de la tabla de voces del arreglador los títulos de las piezas musicales de la población y cualquier otro elemento especificado en la tabla y posteriormente su instrumento asociado.

El proceso de carga de la composición, por otra parte, consiste en rellenar cada fila del arreglador con los títulos que referencian a las muestras musicales de la población. Para visualizar la composición, hacer click en el botón de Actualizar una vez efectuada la carga.

5.7.3 Configuraciones

Para las configuraciones es importante saber qué mutaciones están activadas, qué probabilidad de ocurrencia tienen, y qué tipo de cruce está seleccionado. También es imprescindible conocer el número de individuos y el número de hijos configurados.

Durante la carga de las configuraciones se van leyendo los valores almacenados y una vez recogidos todos, se actualizan en la interfaz, mostrando así la sesión antigua guardada.

Capítulo 6

Conclusiones

Los objetivos iniciales del proyecto eran, como se ha detallado a lo largo del documento, construir una herramienta capaz de:

- Asistir a cualquier tipo de usuario en el proceso de composición.
- Generar nuevos motivos musicales inspirándose en la teoría de los algoritmos genéticos.
- Permitir que sea el usuario quien oriente el tipo de melodías que se van a generar.

Finalizando el proyecto, se puede afirmar que se ha desarrollado de forma satisfactoria una aplicación capaz de llevar a cabo dicho objetivo.

Tras haber usado la aplicación y haber experimentado con ella, llegamos a la conclusión de que, en general, las mutaciones, ya sean aplicadas aleatoriamente o siguiendo un criterio determinado, generan piezas muy variadas y diferentes en sonido en comparación a la música convencionalmente "bonita". Sin embargo, las piezas generadas por estas mutaciones no son por ello menos interesantes. De hecho, han resultado trozos de melodías realmente curiosos que luego han servido para ser añadidos a otras melodías.

Siguiendo en esta línea, las mutaciones que han dado los mejores resultados son aquellas que interpolan los tonos de notas, las que repiten notas, consiguiendo el efecto de patrones repetitivos en las piezas; y aquellas que reducen la duración de las notas, produciendo un efecto más dinámico y vivo, a nuestro juicio.

También hemos comprobado que generando piezas sin mutación alguna, es decir, aplicando únicamente los cruces sobre las piezas, se generan piezas más parecidas a las iniciales. Las piezas resultantes representan variaciones sutiles de las originales y en general, resultan algo más armoniosas en comparación al uso intensivo de mutaciones.

Cabe destacar especialmente los resultados obtenidos gracias al uso del arreglador para mezclar y combinar las piezas. Es en este componente donde hemos podido generar las piezas más interesantes y complejas. La pista de sincronización nos ha resultado fácil e intuitiva de usar, y considerablemente útil para cuadrar y sincronizar las piezas, tal y como deseábamos.

La incorporación del criterio del usuario, inicialmente predicho como un cuello de botella, ha resultado menos tedioso de lo que se esperaba, ya que se configuró el algoritmo de modo que sólo generase hijos de dos en dos, disminuyendo notablemente el tiempo requerido para calificar las piezas. Si bien retrasaba la fase de generación de piezas, el aspecto interactivo derivaba en una situación amena y educativa.

Por otra parte, terceros que han usado la aplicación se han mostrado receptivos con los resultados propuestos por el algoritmo generador y han compuesto piezas en el arreglador sin necesidad de ayuda a la hora de saber cómo proceder. Sin embargo han echado en falta funcionalidades como la opción de poder cambiar ritmos de una pieza automáticamente; un botón de deshacer y sobre todo un repertorio de las posibles modificaciones a hacer a una melodía dentro de la pestaña de edición.

Concluimos que a pesar de no generar piezas notablemente complejas, sí se consigue la música de acompañamiento, *background music*, que se esperaba.

En el enlace se puede escuchar un repertorio de piezas generadas con MyTone:

http://www.youtube.com/user/mytone2013/videos

Capítulo 7

Referencias

[1] Algoritmos Genéticos
http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/temageneticos.pdf

[2] Juan Julián Merelo Guervós, Informática evolutiva: Algoritmos genéticos
http://geneura.ugr.es/~jmerelo/ie/ags.htm

[3] Algoritmo Genético
http://es.wikipedia.org/wiki/Algoritmo_gen%C3%A9tico

[4] Algoritmos Genéticos
http://www.stumptown.com/diss/chapter2.html

[5] Introducción a los algoritmos genéticos y sus aplicaciones, Piedad Tolmos Rodríguez-Piñero
http://www.uv.es/asepuma/X/J24C.pdf

[6] Notación ABC
http://abcnotation.com/about

[7] Notación Musical ABC
http://es.wikipedia.org/wiki/Notaci%C3%B3n_musical_Abc

[8] Música Evolutiva
http://en.wikipedia.org/wiki/Evolutionary_music

[9] Neurogen. Musical composition using genetic algorithms and cooperating neural networks.P. M. Gibson and J. A. Byrne. (1991).
http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=140338

[10] Genetic algorithms and computer-assisted music composition. Horner, A. and Goldberg, D. E. (1991).
http://complex.ccsr.uiuc.edu/web/Techreports/1990-94/CCSR-91-20.pdf

[11] Estudio e implementación de algoritmos genéticos para la generación semi-automática de ritmos y melodías. Enrique Jiménez Domingo. (2009).
http://e-archivo.uc3m.es/bitstream/10016/6069/1/PFC_Enrique_Jimenez_Domingo.pdf

[12] Genjam: A genetic algorithm for generating jazz solos. Biles, J. A. (1994).
http://igm.rit.edu/~jabics/GenJam94/Paper.html

[13] Koan
http://en.wikipedia.org/wiki/Koan_%28program%29

[14] Música Generativa
http://www.intermorphic.com/inmomusic/generative_music.html

[15] Música de fondo. http://en.wikipedia.org/wiki/Background_music

[16] El Proyecto ABC Plus
http://abcplus.sourceforge.net/

[17] TiMidity
http://timidity.sourceforge.net/#info

[18] Ghost4J
http://www.ghost4j.org/index.html

[19] Evolving Four-Part Harmony Using Genetic Algorithm, Patrick Donnelly and John Sheppard. (2011).

http://0-apps.webofknowledge.com.cisne.sim.ucm.es/full_record.do?product=UA&search_mode=GeneralSearch&qid=1&SID=X22inf4FAOde3MDOApP&page=1&doc=1

[20] A Simple Genetic Algorithm for Music Generation by means of Algorithmic Information Theory, Manuel Alfonseca, Manuel Cebrián and Alfonso Ortega.

http://0-ieeexplore.ieee.org.cisne.sim.ucm.es/xpl/articleDetails.jsp?tp=&arnumber=4424858&queryText%3DSimple+Genetic+Algorithm+for+Music+Generation+by+means